Towards Deciding Second-order Unification Problems Using Regular Tree Automata

Tomer Libal

Towards Deciding Second-order Unification Problems Using Regular Tree Automata

Tomer Libal[1]

INRIA
tomer.libal@inria.fr

The second-order unification problem is undecidable [5]. While unification procedures, like Huet's pre-unification, terminate with success on unifiable problems, they might not terminate on non-unifiable ones. There are several decidability results for unification problems with infinitely-many pre-unifiers, such as for monadic second-order problems [3]. These results are based on the regular structure of the solutions of these problems and by computing minimal unifiers.

Beyond the importance of the knowledge that searching for unifiers of decidable problems always terminates, one can also use this information in order to optimize unification algorithms, such as in the case for pattern unification [10].

Nevertheless, being able to prove that the unification problem of a certain class of unification constraints is decidable is far from easy. Some results were obtained for certain syntactic restrictions on the problems (see Levy [8] for some results and references) or on the unifiers (see Schmidt-Schauß [11], Schmidt-Schauß and Schulz [12, 13] and Jeż [7] for some results).

Infinitary unification problems, like the ones we are considering, might suggest that known tools for dealing with the infinite might be useful. One such tool is the regular tree automaton. The drawback of using regular automata for unification is, of course, their inability to deal with variables. In this paper we try to overcome this obstacle and describe an on-going work about using regular tree automata [1] in order to decide more general second-order unification problems.

The second-order unification problems we will consider are of the form $\lambda\overline{z_n}.x_0 t \doteq \lambda\overline{z_n}.C(x_0 s)$ where C is a non-empty context [2] and x_0 does not occur in t or s. We will call such problems *cyclic problems*.

An important result in second-order unification was obtained by Ganzinger et al. [4] and stated that second-order unification is undecidable already when there is only one second-order variable occurring twice. The unification problem they used for proving the undecidability result was an instance of the following cyclic problem. Note that we chose to use in the definition only unary second-order variables but that this restriction should not be essential.

$$x_0(w_1, g(y_1, a)) = g(y_2, x_0(w_2, a)) \tag{1}$$

Our decidability result is obtained by posing one further restriction over cyclic problems which is based on the existence and location of variables other than the cyclic one.

A sufficient condition for the decidability of second-order unification problems was given by Levy [8]. This condition states that if we can never encounter, when applying Huet's pre-unification procedure [6] to a problem, a cyclic equation, then the procedure terminates.

It follows from this result that deciding second-order unification problems depends on the ability to decide cyclic problems. The rules of Huet's procedure (`PUA`) are given in Fig. 1. Imitation partial bindings and projection partial bindings are defined in [14] and are denoted, respectively, by $\mathtt{PB}(f, \alpha)$ and $\mathtt{PB}(i, \alpha)$ where α is a type, Σ a signature $f \in \Sigma$ and $i > 0$.

[1]Funded by the ERC Advanced Grant ProofCert.

$$\frac{S}{S \cup \{A \doteq A\}}\ \texttt{(Delete)} \qquad \frac{S \cup \{\lambda\overline{z_k}.s_1 \doteq \lambda\overline{z_k}.t_1, \ldots, \lambda\overline{z_k}.s_n \doteq \lambda\overline{z_k}.t_n\}}{S \cup \{\lambda\overline{z_k}.f(\overline{s_n}) \doteq \lambda\overline{z_k}.f(\overline{t_n})\}}\ \texttt{(Decomp)}$$

$$\frac{S\sigma \cup \{x \doteq \lambda\overline{z_k}.t\} \qquad x \notin \texttt{FV}(t) \wedge \sigma = [\lambda\overline{z_k}.t/x]}{S \cup \{\lambda\overline{z_k}.x(\overline{z_k}) \doteq \lambda\overline{z_k}.t\}}\ \texttt{(Bind)}$$

$$\frac{S \cup \{x \doteq u, \lambda\overline{z_k}.x^\alpha(\overline{s_n}) \doteq \lambda\overline{z_k}.f(\overline{t_m})\} \qquad u \in \texttt{PB}(f, \alpha)}{S \cup \{\lambda\overline{z_k}.x^\alpha(\overline{s_n}) \doteq \lambda\overline{z_k}.f(\overline{t_m})\}}\ \texttt{(Imitate)}^1$$

$$\frac{S \cup \{x \doteq u, \lambda\overline{z_k}.x^\alpha(\overline{s_n}) \doteq \lambda\overline{z_k}.a(\overline{t_m})\} \qquad i > 0 \leq k, u = \texttt{PB}(i, \alpha)}{S \cup \{\lambda\overline{z_k}.x^\alpha(\overline{s_n}) \doteq \lambda\overline{z_k}.a(\overline{t_m})\}}\ \texttt{(Project)}^2$$

1. where $f \in \Sigma$.
2. where either $a \in \Sigma$ or $a = z_i$ for some $0 < j \leq k$.

Figure 1: PUA- Huet's pre-unification procedure

In order to describe the algorithm, we give next several technical definitions. These definitions are taken from our previous work on extending PUA to deal with some non-termination [9]. The discussions there can help to supplement the description of the definitions given here.

The main technique behind the results of this paper is to analyze the application of PUA on cyclic equations. A crucial component for obtaining non-termination is the generation of new cyclic equations in each step. The following definition of progressive contexts describes the form of these new cyclic equations along the execution of PUA.

Let e be a cyclic equation as above where $C = C_1 \ldots C_m$ such that for all $0 < i \leq m$, $C_i = f_i(r_i^1, \ldots, [.], \ldots, r_i^{n_i})$ where $n_i = \texttt{arity}(f_i) - 1$. Define also, for all $m < i$, $C_i = f_k(y_{i-m}^1 s, \ldots, [.], \ldots, y_{i-m}^{n_k} s)$ where $k = ((i-1) \mod m) + 1$ and y_{i-m}^j for $0 < j \leq n_k$ are new variables. We define the progressive context D_i^e for all $0 \leq i$ as $D_i^e = C_{i+1} \ldots C_{i+m}$.

In the rest of this paper, e will refer to equations of this form and $t, s, C, m, k, n_i, r_i^j$ and y_i^j will refer to the corresponding values in e.

In order to clarify the definitions, we will use the following (non-unifiable) cyclic equation as an example: $x_0 f(a, a) \doteq f(x_0 a, f(f(a, a), b))$. Note that PUA does not terminate on this problem.

For the example, $t = f(a, a), s = a, C = f([.], f(f(a, a), b)), m = 1, k = 1, n_i = 1$, and $r_i^1 = f(f(a, a), b)$ for all $0 < i$. The progressive contexts for this example are: $D_0 = C, D_1 = f([.], y_1 a), D_2 = f([], y_2 a)$, etc.

Since the only "don't know" non-determinism in PUA is due to the choices in the search between the rules (Imitate) and (Project) [14], we can follow the execution of PUA on the cyclic equation e using the search tree in Figure 2 and using the following definitions. Given a cyclic equation e, for all $0 \leq i$, we define $\mathfrak{I}(i)$, $\mathfrak{I}^*(i)$ and $\mathfrak{P}(i)$ inductively as follows:

- $\mathfrak{P}(0) = \mathfrak{I}(0) = \mathfrak{I}^*(0) = \emptyset$.
- if $0 < i \leq m$ then $\mathfrak{I}^*(i) = \mathfrak{I}^*(i-1) \cup \{\lambda\overline{z_n}.y_i^j t \doteq \lambda\overline{z_n}.r_i^j \mid 1 \leq j \leq n_i\}$.
- if $m < i$ then $\mathfrak{I}^*(i) = \mathfrak{I}^*(i-1) \cup \{\lambda\overline{z_n}.y_i^j t \doteq \lambda\overline{z_n}.y_{i-m}^j s \mid 1 \leq j \leq n_i\}$.
- for all $i > 0$, $\mathfrak{I}(i) = \mathfrak{I}^*(i) \cup \{\lambda\overline{z_n}.x_i t \doteq \lambda\overline{z_n}.D_i^e(x_i s)\}$.
- for all $i > 0$, $\mathfrak{P}(i) = \mathfrak{I}^*(i-1) \cup \{\lambda\overline{z_n}.t \doteq \lambda\overline{z_n}.D_{i-1}^e(s)\}$.

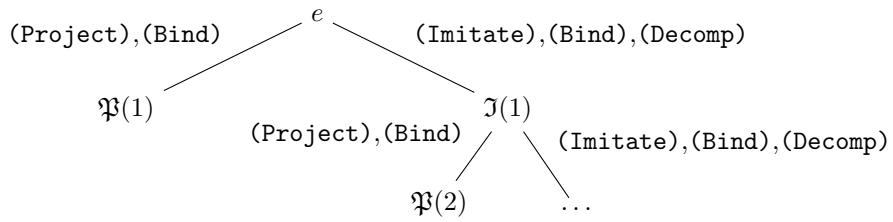

Figure 2: The "don't-know" non-determinism in PUA

As was noted above, the progressive context is being used in order to describe the form of the new cyclic equations in the sets $\mathfrak{P}(i)$ and $\mathfrak{I}(i)$.

From the fact that PUA is complete for higher-order unification [6] and from the fact that e is a cyclic problem, it follows that e is unifiable iff there is $i > 0$ such that $\mathfrak{P}(i)$ is unifiable [9].

For the example, we have $\mathfrak{P}(1) = \{f(a,a) \doteq f(f(a,a),b)\}$, $\mathfrak{P}(2) = \{y_1 f(a,a) \doteq f(f(a,a),b), f(a,a) \doteq f(a,y_1 a)\}$, $\mathfrak{P}(3) = \{y_1 f(a,a) \doteq f(f(a,a),b), y_2 f(a,a) = y_1 a, f(a,a) \doteq f(a,y_2 a)\}$, etc.

Zaionc [15] gave a finite description of complete sets of unifiers for some monadic second-order problems by proving that a structure based on the matching tree generated by Huet's procedure [6] is regular. We would like to pursue a similar approach for our cyclic problems but unlike the monadic case, where each node in the matching tree contains the same number of equations, for non-monadic signatures there is no bound on the number of equations in the nodes.

In the following definition we remove equations from the $\mathfrak{P}$ sets in order to obtain similar regularity.

Let $\mathfrak{P}^-(i) \subseteq \mathfrak{P}(i)$ be the following set of equations.

- if $0 < i \leq m+1$ then $\mathfrak{P}^-(i) = \mathfrak{P}(i)$.
- if $m+1 < i$ then $\mathfrak{P}^-(i) = \mathfrak{I}^*(m) \cup \{\lambda\overline{z_n}.t \doteq \lambda\overline{z_n}.D^e_{i-1}(s)\}$.

and let $\mathfrak{P}^*(i) = \mathfrak{P}(i) \setminus \mathfrak{P}^-(i)$.

The corresponding values for our example are $\mathfrak{P}^-(1) = \mathfrak{P}(1), \mathfrak{P}^-(2) = \mathfrak{P}(2)$, $\mathfrak{P}^-(3) = \{y_1 f(a,a) \doteq f(f(a,a),b), f(a,a) \doteq f(a,y_2 a)\}$ and $\mathfrak{P}^*(3) = \{y_2 f(a,a) = y_1 a\}$.

Clearly, $\mathfrak{P}(i)$ is unifiable only if both $\mathfrak{P}^-(i)$ and $\mathfrak{P}^*(i)$ are (using the same substitution). In [9] we proved that $\mathfrak{P}^-(i)$ is unifiable only if there is $0 < j \leq 3m \leq i$ such that $\mathfrak{P}^-(j)$ is unifiable. I.e. that we can always decide if there is an $i > 0$ such that $\mathfrak{P}^-(i)$ is unifiable.

In the present paper, we will describe how to decide the unification problem of these cycles under a very strong restriction. We require that the generated sets $\mathfrak{P}^-$ have at most finitely-many ground unifiers. This is a very strong requirement since even simple first-order problems like $f(x) = y$ have infinitely-many ground unifiers.. Nevertheless, we hope that this preliminary work can be extended to stronger classes. A class subsumed by the above and describable syntactically is the class of second-order problems containing one variable (second or first-order) occurring at most twice. At the same time, the class described by the cyclic problems in Equation 1 and which was shown to be undecidable [4] may generate $\mathfrak{P}^-$ sets having infinitely-many most general unifiers.

In order to decide if such cyclic equations are unifiable, we will first investigate the equations in $\mathfrak{P}^-(i)$ and $\mathfrak{P}^*(i)$ and the relation between them.

Let us consider $\mathfrak{P}(i)$ for some $i > 0$ and let us pick an arbitrary equation $\lambda\overline{z_{n_1}}.y_k^j t \doteq \lambda\overline{z_{n_1}}.r_k^j \in \mathfrak{P}^-(i)$ where $0 < k \leq m$ and j is some index depending on the arity of the enclosing function symbol in C. This equation is connected to the following set of equations in $\mathfrak{P}^*(i)$: $\{\lambda\overline{z_{n_2}}.y_{k+m}^j t \doteq \lambda\overline{z_{n_2}}.y_k^j s, \lambda\overline{z_{n_3}}.y_{k+2m}^j t \doteq \lambda\overline{z_{n_3}}.y_{k+m}^j s, \ldots, \lambda\overline{z_{n_l}}.y_{\frac{i-1}{m}}^j t \doteq \lambda\overline{z_{n_l}}.y_{\frac{i-1}{m}-m}^j s\}$. The last occurrence of a variable in this chain is the occurrence of $y_{\frac{i-1}{m}}^j s$ in the equation $\lambda\overline{z_{n_{l+1}}}.t \doteq \lambda\overline{z_{n_{l+1}}}.D_{i-1}^e(s) \in \mathfrak{P}^-(i)$. Call the equations from $\mathfrak{P}^-(i)$ base equations and the ones from $\mathfrak{P}^*(i)$ inductive equations. In the following algorithm we will consider first the finitely-many $\mathfrak{P}(j)$ problems for $0 < j \leq 3m$ and will then consider the problems $\mathfrak{P}(i)$ such that $\mathfrak{P}^-(i)$ is isomorphic to $\mathfrak{P}^-(j)$ (this is determined according to the indices i and j, see [9] for more information). We will call the infinitely-many such problems $\mathfrak{P}(i)$ the extensions of $\mathfrak{P}(j)$.

Since in our example m equals 1, there can be only one chain. For $\mathfrak{P}(i)$ the chain is just the sequence of equations in $\mathfrak{P}(i)$.

We will require another restriction for the remaining part of the paper and will consider only the case when there is one chain and renumber the indices of the y variables with $1, \ldots, p$. We believe that both this restriction and our use of unary cyclic variables are not essential to the results obtained. Since there is only one chain and the chain and the problem $\mathfrak{P}(i)$ are the same, we will consider both the problems $\mathfrak{P}(i)$ as extensions of $\mathfrak{P}(j)$ and the (single) chains of $\mathfrak{P}(i)$ as extensions to the chain in $\mathfrak{P}(j)$.

In order to define the algorithm, we need first to define how to construct the regular tree automaton based on three terms. The first term will be of the form $\lambda z.u$ where u can contains z but no subterm of the form v, the second term will be v and the third term will be w such that w contains occurrences of z and has no subterm of the form v. Using these three terms, we define the following tree automaton $A = (Q, Q_f, \Delta)$ where $Q = \{q_w, q_u\}, Q_f = \{q_u\}$ and Δ is defined as follows:

- $\lambda z.u_1 \to q_u(\lambda z.u_2)$ where u_1 is obtained from u by replacing each occurrence of z with $q_w(x_l)$ where l is a new index for each occurrence. u_2 is obtained in the same way but we replace each occurrence of z with x_l.
- $w_1 \to q_w(w_2)$ where w_1 and w_2 are obtained from w in the same way u_1 and u_2 were obtained from u,
- $v \to q_w(v)$.
- $z \to q_w(z)$.

We describe this automaton using $\mathtt{aut}(u, v, w)$.

The idea behind this construction is that the language accepted by this automaton is exactly the one containing all the possible mappings for y_1 in the chain according to the arbitrarily-many constraints in the inductive part of the chain and based on a given unifier σ for the base part of the chain. Given this automaton, we need just to test if $\sigma(y_1)$ is recognized by it. Note that the base equations contain only occurrences of the variables y_1 and y_p and therefore, a unifier for the base part poses no constraints on the values for the variables $y_2, \ldots, y_{p-1}$ in unifiers for the inductive part and we can freely generate all possible substitutions.

We describe next an algorithm for deciding our cyclic problems.

1. given a cyclic problem e.
2. compute the (finite) complete set of unifiers for some $\mathfrak{P}^-(j)$ where $0 < j \leq 3m$.
3. let σ be such a unifier and for the single chain in $\mathfrak{P}^-(j)$ do the following:

(a) let $\lambda z.u$ be obtained from $\sigma(y_p)$ by replacing all occurrences of $s\sigma$ with z..

(b) let $v = s\sigma$.

(c) let w be obtained from $t\sigma$ by replacing all occurrences of $s\sigma$ in it with z.

(d) fail if $\sigma(y_1)$ is not recognized by $\mathtt{aut}(u, v, w)$.

The algorithm tries to find one unifier of a $\mathfrak{P}^-(j)$ which can be extended in order to unify some extension $\mathfrak{P}^*(i)$ of $\mathfrak{P}^*(j)$ for an arbitrary $i > 0$.

The correctness of the above algorithm is based on the following theorem.

Theorem 1. Given a problem e, $0 < j \leq 3m$, a chain in $\mathfrak{P}(j)$ over variables $y_1, \ldots, y_p$ and a unifier σ of $\mathfrak{P}^-(j)$, $\sigma(y_1)$ is recognized by $\mathtt{aut}(u, v, w)$ as above iff there is an extension $\mathfrak{P}(i)$ of $\mathfrak{P}(j)$ and a substitution θ, such that θ unifies $\mathfrak{P}(i)$.

Proof. Proof sketch: the automata for the chain and all its extensions are the same.

- if - by induction on the number of the variables $y_1, \ldots, y_q$ in the extension $\mathfrak{P}(i)$ of $\mathfrak{P}(j)$. For the step we need to prove that the first equation in the chain (which is determined last since the base of the terms in the language are determined by $\theta(y_q)$) is unifiable only if $\sigma(y_1)$ is recognized by the automaton. Since θ extends σ, we have $\theta(y_1) = \sigma(y_1)$ and by assuming that $\theta(y_2)$ is recognized, the rest follows from the definition of the automaton.

- only if - for this direction, we need to choose the extension $\mathfrak{P}(i)$ and build the substitution θ. This is computed by considering the accepting sequence of transitions for $\sigma(y_1)$. The maximal number of nested transitions determines the number of equations in the chain while the transitions themselves determines the values $\theta(y_2), \ldots, \theta(y_{q-1})$ in the chain. In addition, $\theta(y_1) = \sigma(y_1)$ and $\theta(y_q) = \sigma(y_p)$.

□

We will now demonstrate this idea on the example. A unifier σ for $\mathfrak{P}^-(2)$ is $[y_2 \mapsto \lambda z.z, y_1 \mapsto \lambda z.f(z, b)]$. Using the unifier we get that $u = \lambda z.z, v = a$ and $w = f(z, z)$. Therefore, $\mathtt{aut}(u, v, w) = (Q, Q_f, \Delta)$ where $Q = \{q_w, q_u\}, Q_f = \{q_u\}$ and Δ is defined as follows:

- $\lambda z.q_w(x) \to q_u(\lambda z.x)$.
- $f(q_w(x_1), q_w(x_2)) \to q_w(f(x, y))$.
- $a \to q_w(a)$.
- $z \to q_w(z)$.

Clearly $\sigma(y_1)$ cannot be generated by $\mathtt{aut}(u, v, w)$ and therefore we have proved, according to the previous theorem, that there is no possible extension of σ. By doing the same to all unifiers of $\mathfrak{P}^-(1), \mathfrak{P}^-(2)$ and $\mathfrak{P}^-(3)$, we can prove that the example is not unifiable.

To summarize, we have described a decision algorithm for restricted cyclic second-order unification problems. Although the class is strongly restricted, it includes the class of second-order problems containing only one variable (first or second-order) which occurs at most twice. The main novelty of the method is the use of tree automata in order to decide unification problems. At the same time, the exact syntactic form of cycles in this class and the possibility to treat problems with more than one chain are still under investigation.

References

[1] H Comon, M Dauchet, R Gilleron, F Jacquemard, D Lugiez, S Tison, and M Tommasi. Tree automata techniques and applications, 1999.

[2] Hubert Comon. Completion of rewrite systems with membership constraints. part i: Deduction rules. *J. Symb. Comput.*, 25(4):397–419, 1998.

[3] William M. Farmer. A unification algorithm for second-order monadic terms. *Annals of Pure and Applied Logic*, 39(2):131–174, 1988.

[4] Harald Ganzinger, Florent Jacquemard, and Margus Veanes. Rigid reachability. In *Proceedings of the 4th Asian Computing Science Conference on Advances in Computing Science (ASIAN 98)*, volume 1538 of *LNCS*, pages 4–21. Springer Verlag, 1998.

[5] Warren D. Goldfarb. The undecidability of the second-order unification problem. *Theor. Comput. Sci.*, 13:225–230, 1981.

[6] Gérard P. Huet. A unification algorithm for typed lambda-calculus. *Theor. Comput. Sci.*, 1(1):27–57, 1975.

[7] Artur Jez. Context unification is in PSPACE. In *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part II*, pages 244–255, 2014.

[8] Jordi Levy. Decidable and undecidable second-order unification problems. In *RTA*, pages 47–60, 1998.

[9] Tomer Libal. Regular patterns in second-order unification. 2015. to appear. http://logic.at/staff/shaolin/papers/holunif.pdf.

[10] Dale Miller. Unification of simply typed lambda-terms as logic programming. In *In Eighth International Logic Programming Conference*, pages 255–269. MIT Press, 1991.

[11] Manfred Schmidt-Schauß. A decision algorithm for stratified context unification. *J. Log. Comput.*, 12(6):929–953, 2002.

[12] Manfred Schmidt-Schauß and Klaus U. Schulz. Solvability of context equations with two context variables is decidable. *J. Symb. Comput.*, 33(1):77–122, 2002.

[13] Manfred Schmidt-Schauß and Klaus U. Schulz. Decidability of bounded higher-order unification. *J. Symb. Comput.*, 40(2):905–954, August 2005.

[14] Wayne Snyder and Jean H. Gallier. Higher-order unification revisited: Complete sets of transformations. *J. Symb. Comput.*, 8(1/2):101–140, 1989.

[15] Marek Zaionc. The regular expression descriptions of unifier sets in the typed lambda calculus. In *Fundamenta Informaticae X*, pages 309–322. North-Holland, 1987.